Neural Networks 122 (2020) 218–230

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A complementary learning systems approach to temporal difference learning

Sam Blakeman *, Denis Mareschal

Centre for Brain and Cognitive Development, Department of Psychological Sciences, Birkbeck, University of London, Malet Street, WC1E 7HX, United Kingdom

ARTICLE INFO

Article history:
Received 8 May 2019
Received in revised form 21 August 2019
Accepted 17 October 2019
Available online 26 October 2019

Keywords:
Complementary learning systems
Reinforcement learning
Hippocampus

ABSTRACT

Complementary Learning Systems (CLS) theory suggests that the brain uses a 'neocortical' and a 'hippocampal' learning system to achieve complex behaviour. These two systems are complementary in that the 'neocortical' system relies on slow learning of distributed representations while the 'hippocampal' system relies on fast learning of pattern-separated representations. Both of these systems project to the striatum, which is a key neural structure in the brain's implementation of Reinforcement Learning (RL). Current deep RL approaches share similarities with a 'neocortical' system because they slowly learn distributed representations through backpropagation in Deep Neural Networks (DNNs). An ongoing criticism of such approaches is that they are data inefficient and lack flexibility. CLS theory suggests that the addition of a 'hippocampal' system could address these criticisms. In the present study we propose a novel algorithm known as Complementary Temporal Difference Learning (CTDL), which combines a DNN with a Self-Organizing Map (SOM) to obtain the benefits of both a 'neocortical' and a 'hippocampal' system. Key features of CTDL include the use of Temporal Difference (TD) error to update a SOM and the combination of a SOM and DNN to calculate action values. We evaluate CTDL on Grid World, Cart–Pole and Continuous Mountain Car tasks and show several benefits over the classic Deep Q-Network (DQN) approach. These results demonstrate (1) the utility of complementary learning systems for the evaluation of actions, (2) that the TD error signal is a useful form of communication between the two systems and (3) that our approach extends to both discrete and continuous state and action spaces.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 1998) represents a computational framework for modelling complex reward-driven behaviour in both artificial and biological agents. For cognitive scientists it is of continuing interest to explore how RL theory maps onto neural structures in the brain (Lee, Seo, & Jung, 2012; Niv, 2009). One of the most influential findings in this regard is the encoding of Temporal Difference (TD) error by phasic midbrain dopaminergic neurons (Schultz, 2016; Schultz, Dayan, & Montague, 1997). One of the major projection sites of these neurons is the striatum (Bray & Doherty, 2007; Doherty, Dayan, Friston, Critchley, & Dolan, 2003; Mcclure, Berns, & Montague, 2003) and it has been proposed that the striatum is responsible for evaluating states and actions for decision making (Houk, Adams, & Barto, 1995; Roesch, Singh, Brown, Mullins, & Schoenbaum, 2009; Schultz, 1998; Schultz, Apicella, Scarnati, & Ljungberg, 1992; Setlow, Schoenbaum, & Gallagher, 2003). Interestingly, the striatum receives inputs from both cortical areas and the hippocampus, suggesting that it is responsible for evaluating different forms of information.

Complementary Learning Systems (CLS) theory posits that the neocortex and hippocampus have complementary properties that allow for complex behaviour (Kumaran, Hassabis, & McClelland, 2016; McClelland, McNaughton, & O'Reilly, 1995). More specifically, the hippocampus relies on fast learning of conjunctive, pattern-separated memories. These memories then support the learning of a second system, the neocortex, which slowly learns distributed representations that support generalization across features and experiences. The purpose of the present study is to explore how the brain's RL machinery might utilize these opposing properties to achieve complex behaviour.

Much of the recent success of RL has been due to the combination of classical RL approaches with the function approximation properties of Deep Neural Networks (DNNs), known as deep RL (François-lavet et al., 2018). Typically in deep RL, the action-value function $Q(s, a)$ is represented using a DNN that takes the state s_t as input and outputs the corresponding action values

* Corresponding author.
E-mail addresses: sblake03@mail.bbk.ac.uk (S. Blakeman), d.mareschal@bbk.ac.uk (D. Mareschal).

https://doi.org/10.1016/j.neunet.2019.10.011

for that state. Despite impressive results, such as super human-level performance on Atari video games (Mnih et al., 2015), deep RL approaches are often criticized for being data inefficient and adapting poorly to changes in the input distribution (Lake, Ullman, Tenenbaum, & Gershman, 2017). From a CLS perspective, the DNNs used in deep RL can be seen as sharing many commonalities with a 'neocortical' learning system. In particular, both the neocortex and DNNs rely on small learning rates and distributed representations for efficient generalization.

CLS theory suggests that the addition of a 'hippocampal' learning system to deep RL approaches may improve our understanding of how RL is implemented in the brain and address the aforementioned criticisms of deep RL. Ideally an agent should be able to utilize the advantages of a 'neocortical' system (distributed representations and generalization) and a 'hippocampal' system (fast learning and pattern-separation) to perform complex reward-driven behaviour. Indeed, many theoretical advantages have been proposed for the use of hippocampal episodic information in RL. In particular, it has been suggested that episodic information can be used to approximate value functions, increase data efficiency and reconcile long-range dependencies (Gershman & Daw, 2017).

An alternative to using a DNN to represent the action-value function is to represent it in a tabular manner (Sutton & Barto, 1998). Such an approach is more in line with a hippocampal learning system as experiences are stored in a pattern-separated manner and larger learning rates can be used. Importantly, the tabular case means that every action value is stored as its own memory, which eliminates the potential for interference. This is in contrast to DNNs that naturally suffer from interference due to their distributed representations. However, as the number of states and/or actions increases, the tabular case will require more experience to encounter each action-value and more computational resources to store the values. The distributed representations of DNNs then become advantageous because they allow for efficient generalization over the state space. In an ideal scenario a DNN would be responsible for generalization over certain areas of the state space while a tabular method would store pattern-separated memories that are crucial to behaviour and that violate the generalizations of the network.

Previous work in deep RL has often touched upon CLS theory and the benefits of a hippocampal learning system. Indeed, one of the most influential deep RL approaches, the Deep Q-Network (DQN) (Mnih et al., 2015), utilizes a secondary system that tentatively mirrors a hippocampal learning system. More specifically, the DQN has a table that stores past experiences in a pattern-separated manner and then uses them to train a DNN in an interleaved fashion. This process is tentatively compared to 'replay'; a biological phenomenon that appears to replay information from the hippocampus to the neocortex in biological agents (Olafsdottir, Bush, & Barry, 2018). However, despite the DQN having what appears to be two complementary learning systems, the decision making (calculation of Q values) is ultimately based on the predictions of the DNN, which learns slowly via distributed representations.

More recently, research in deep RL has begun to demonstrate some of the advantages of an explicit 'hippocampal' learning system that evaluates states and actions (Botvinick et al., 2019). Most notably (Blundell, Pritzel, & Rae, 2016) proposed an algorithm called 'model-free episodic control', which consisted of a table containing the maximum return (sum of discounted rewards) for each state–action pair experienced. The memory requirements for this table were kept constant by removing the least recently updated table entry once the size limit had been reached. Each observation from the environment was projected by an embedding function (either a random projection or a variational autoencoder) to a state value and actions were selected based on a k-nearest neighbours method, which allowed for some degree of generalization to novel states. Blundell et al. (2016) tested this approach on the Arcade Learning Environment (Atari) (Bellemare, Naddaf, Veness, & Bowling, 2013) and Labyrinth (Mnih et al., 2016), which both require the use of visual information to learn an optimal policy. The results of these simulations showed that model-free episodic control was significantly more data efficient than other classical deep RL approaches, suggesting that episodic information is important for fast learning.

While taking a first step towards highlighting the benefits of a 'hippocampal' learning system that utilizes fast learning of pattern-separated information, the work of Blundell et al. (2016) has several notable drawbacks. Firstly, the table recorded the maximum return from any given episode and used this to inform the policy of the agent. This naturally cannot handle stochastic environments, where the expected return is the important quantity and not the maximum return of an individual episode. Secondly, this approach is likely to be highly inflexible. For example if a state–action pair suddenly becomes highly aversive then the entry in the table will not be updated because only the maximum value is stored. A third criticism is that the approach relies on the full return for each state–action pair and this is only possible when the task has distinct finite episodes. Some of these criticisms have been addressed in subsequent work, for example Pritzel et al. (2017) propose a fully differentiable version of 'model-free episodic control' that learns the embedding function in an online fashion using N-step Q-learning.

The above issue not withstanding, what is most pertinent to the present study is that 'model-free episodic control', and its various derivatives, do not rely on two complementary learning systems that operate in parallel to evaluate actions. The embedding function may be tentatively compared to a 'neocortical' learning system but it operates before the 'hippocampal' learning system and as a result only the output of the 'hippocampal' learning system is used to evaluate action values. This means that any advantages that may be conferred from the additional predictions of a 'neocortical' learning system are lost. In essence, the aforementioned approaches cannot arbitrate between the predictions of a 'neocortical' and a 'hippocampal' learning system, but are instead restricted to using episodic predictions. This is inconsistent with the finding that the striatum receives inputs from both cortical areas and the hippocampus and needs to arbitrate between the two (Pennartz, Ito, Verschure, Battaglia, & Robbins, 2011).

With these criticisms in mind, we present a novel method for imbuing a deep RL agent with both a 'neocortical' and a 'hippocampal' learning system so that it benefits from both types of learning system. Most importantly these two systems: (1) learn in parallel, (2) communicate with each other using a biologically plausible signal, and (3) both make action value predictions. We represent the 'neocortical' system as a DNN and the 'hippocampal' system as a Self-Organizing Map (SOM). Importantly, the size of the SOM is significantly smaller than the state space experienced by the agent so as to replicate the restricted computational resources experienced by biological agents. The SOM is tasked with storing pattern-separated memories of states that the DNN is poor at evaluating. To achieve this we use the TD error from a DNN in order to train the SOM. Critically, this novel CLS approach demonstrates how the TD error of a 'cortical' system can be used to inform a 'hippocampal' system about when and what memories should be stored, with both systems contributing to the evaluation of action-values. This allows the agent to utilize the benefits of both a neocortical and hippocampal learning system for action selection. We call our novel algorithm *Complementary Temporal Difference Learning* (CTDL) and demonstrate that it can improve the performance and robustness of a deep RL agent on Grid World, Cart–Pole and Continuous Mountain Car tasks.

2. Methods

2.1. The reinforcement learning problem

The general goal of a reinforcement learning agent is to select actions based on perceived states in order to maximize future expected rewards. In simple terms the agent chooses an action a_t given a state s_t. The environment then responds to this decision and produces the next state s_{t+1} and a reward r_{t+1}. This agent-environment loop continues either indefinitely or until a terminal state is reached. Typically future rewards are discounted so that more immediate rewards are worth more than distant rewards. This is done using a discount factor $\gamma \in (0, 1)$ which is applied at each time step. The return R_t at time t is defined as the discounted sum of future rewards:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{1}$$

where $r_{t'}$ is the reward value at time t' and T is the time-step at which the task or episode finishes. Solving the RL problem equates to learning a policy π that maps from states to actions ($\pi : s \mapsto a$) and achieves the greatest possible expected return from every state. This is known as the optimal policy π^*. One possible method of finding π^* is to learn the optimal action-value function $Q^*(s, a)$, which provides the expected return of taking action a in state s and following π^* thereafter.

$$\begin{aligned} Q^*(s, a) &= \max_{\pi} Q^{\pi}(s, a) \\ &= \max_{\pi} \mathbb{E}_{\pi}[R_t \mid s_t = s, a_t = a] \end{aligned} \tag{2}$$

Once an agent has learnt the optimal action-value function it can act optimally by picking the action with the largest Q value given the state it is in ($\arg\max_a Q(s, a)$). Importantly the optimal action-value function can be defined recursively using the Bellman equation:

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') | s, a] \tag{3}$$

This recursive definition of the action-value function forms the basis for many learning algorithms. One such algorithm is Q-learning, which is a form of Temporal Difference (TD) learning. Q-learning utilizes the following update rule to learn the optimal action-value function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{4}$$

where $[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$ is known as the TD error, which has been proposed to exist in biological agents (Doherty et al., 2003; Schultz et al., 1997). Importantly the action-value function can be represented in a tabular manner or with a function approximator such as a DNN. In the case of a DNN, the state is provided as input and each output unit corresponds to the value of a single action. The parameters of the network θ are updated so that $Q(s, a)$ moves closer to $r + \gamma \max_a Q(s_{t+1}, a)$. In order to achieve this the objective function of the neural network is set to the mean squared error between the two values i.e. the mean square of the TD error:

$$J(\theta) = \mathbb{E}_{s_t, a_t, r_{t+1}, s_{t+1}}[((r_{t+1} + \gamma \max_{a} Q(s_{t+1}, a; \theta)) - Q(s_t, a_t; \theta))^2] \tag{5}$$

2.2. Complementary Temporal Difference Learning (CTDL)

Our novel approach combines a DNN with a SOM to imbue an agent with the benefits of both a 'neocortical' and 'hippocampal' learning system. The DNN is a simple feed-forward network that takes the current state as input and outputs the predicted action values for each action. The network is trained using the same training objective as Mnih et al. (2015) and a copy of the network is made every C time steps in order to improve training stability. The optimizer used was RMSProp and the hyper-parameter values can be seen in Table 2. Importantly, unlike in Mnih et al. (2015), no memory buffer is used to record past experiences, which saves considerable memory resources. The SOM component is represented as a square grid of units, with each unit having a corresponding action-value $Q(u, a)$ and weights β_u that represent a particular state.

Table 1
Grid world hyper-parameter values used for all simulations.

Parameter	Value	Description
W	10	Width of grid world
H	10	Height of grid world
E	1000	Number of episodes for learning
T	1000	Maximum number of time steps per episode

A general outline of the algorithm detailing how the DNN and SOM interact can be seen in Algorithm 1. In simple terms, the TD error produced by the DNN is used to update the SOM and both systems are used to calculate Q values for action selection. When the agent observes the state s_t, the closest matching unit in the SOM u_t is calculated based on the euclidean distance between the units weights β_u and s_t. This distance is also used to calculate a weighting parameter $\eta \in \{0, 1\}$, which is used to calculate a weighted average of the action values from the SOM and the DNN. If the best matching unit is close to the current state then a larger weighting will be applied to the Q value produced by the SOM. A free parameter τ_η acts as a temperature parameter to scale the euclidean distance between β_u and s_t when calculating the weighted average.

For learning in both the DNN and the SOM, the TD error is calculated using the difference between the target value and the predicted Q value of the DNN. The TD error is used to perform a gradient descent step with respect to the parameters θ of the DNN, which ensures that the predictions of the DNN move towards the weighted average of the SOM and DNN predictions. After updating the DNN, the TD error is also used to update the SOM. More specifically, the TD error is used to create an exponentially increasing value $\delta \in \{0, 1\}$, which scales the standard deviation of the SOM's neighbourhood function and the learning rate of the SOM's weight update rule. Again a temperature parameter τ_δ is used to scale the TD error. Next, the action value of the closest matching unit from the previous time step $Q^{SOM}(u_{t-1}, a_{t-1})$ is updated using the learning rate ρ, the weighting from the previous time step η_{t-1} and the difference between $Q^{SOM}(u_{t-1}, a_{t-1})$ and the target value y_t. The inclusion of η_{t-1} ensures that the action value only receives a large update if the closest matching unit is similar to the state value.

To aid in the training of the DNN and to mimic biological 'replay', the contents of the SOM are replayed to the DNN as a training batch for gradient descent. To construct the training batch the actions a_t are sampled randomly, the states s_t are set to a random sample of the SOM weights β_u and the target values y_t are set to the corresponding Q values stored in the SOM. Finally, the agent's actual action is chosen in an ϵ-greedy manner with respect to the weighted average of the predicted DQN and SOM Q values.

The aforementioned algorithm has several interesting properties. Firstly, the calculation of Q values involves the contribution of both the DNN and the SOM. The size of their respective contributions are controlled by the parameter η, which ensures that

Algorithm 1 - CTDL. Highlighted lines are unique to CTDL when compared to DQN

Initialize probability of selecting a random action $\epsilon = 1$
Initialize SOM weights β to random locations in the grid world
Initialize SOM action-values $Q^{SOM} = 0$
Initialize action-value function Q^{DNN} with random weights θ
Initialize target action-value function $\tilde{Q}^{DNN}$ with weights $\theta^- = \theta$
for $e = 1, E$ **do**
  If $\epsilon > \epsilon^{end}$ then decrease ϵ by $\epsilon^{end}/E^{\epsilon}$
  for $t = 1, T$ **do**
    Observe current state s_t and reward r_t
    Retrieve SOM unit u_t that is closest to s_t
    $u_t = \arg\min_u ||\beta_u - s_t||^2$
    Calculate weighting η based on distance
    $\eta = \exp(-||\beta_{u_t} - s_t||^2/\tau_\eta)$
    Calculate $Q(s_t, a')$ as weighted average of SOM and DNN values
    $Q(s_t, a') = \eta Q^{SOM}(u_t, a') + (1 - \eta)\tilde{Q}^{DNN}(s_t, a'; \theta^-)$

$$\text{set } y_t = \begin{cases} r_t & \text{if episode is over} \\ r_t + \gamma \max_{a'} Q(s_t, a') & \text{otherwise} \end{cases}$$

    Perform gradient descent step on $(y_t - Q^{DNN}(s_{t-1}, a_{t-1}; \theta))^2$ with respect to the network parameters θ
    Calculate δ based on the TD error produced by the DNN
    $\delta = \exp(|y_t - Q^{DNN}(s_{t-1}, a_{t-1}; \theta)|/\tau_\delta) - 1$
    Calculate the neighbourhood function based on u_{t-1}
    $T_{u_j, u_{t-1}} = \exp(-||l_{u_j} - l_{u_{t-1}}||^2/2(\sigma_c + (\delta * \sigma)))$
    Update the weights β of SOM
    $\Delta\beta_{ji} = \alpha * \delta * T_{u_j, u_{t-1}}(s_{t-1,i} - \beta_{ji})$
    Update the action value $\Delta Q^{SOM}(u_{t-1}, a_{t-1}) =$
    $\rho * \eta_{t-1} * (y_t - Q^{SOM}(u_{t-1}, a_{t-1}))$
    Replay contents of SOM to DNN using a random sample of actions
    a_t and unit weights β_u. y_t is set to $Q^{SOM}(\beta_u, a_t)$
    Select random action with probability ϵ, else $a_t =$
    $\arg\max_{a'} \eta Q^{SOM}(u_t, a') + (1 - \eta)Q^{DNN}(s_t, a'; \theta)$
    Every C steps reset $\tilde{Q}^{DNN} = Q^{DNN}$
    If the goal has been reached then **break** and end episode
  end for
end for

if the current state is close to one stored in SOM memory then the Q value predicted by the SOM will have a larger contribution. This is akin to retrieving a closely matching episodic memory and using its associated value for action selection. Secondly, because the SOM is updated using the TD error produced by the DNN, it is biased towards storing memories of states that the DNN is poor at evaluating. Theoretically this should allow the DNN to learn generalizations across states, while the SOM picks up on violations or exceptions to these generalizations and stores them in memory along with a record of their action values. If after many learning iterations the DNN converges to a good approximation of the optimal action-value function then no TD error will be produced and the SOM will be free to use its resources for other tasks. Finally, the SOM can use much larger learning rates than the DNN because it relies on a tabular approximation of the action-value function, which should improve data efficiency.

2.3. Simulated environments

2.3.1. Grid world task

To evaluate our approach we generated a set of symmetric 2D grid worlds (Fig. 1). Each cell in the grid world represents a state $s \in \mathbf{R}^2$ that is described by its x and y position. If N is the number of cells in the grid world, then $\frac{N}{5}$ negative rewards (-1) are randomly placed in the grid world along with a single positive reward $(+1)$ and the agents starting position. The agent's task is to reach the positive reward, at which point the episode is over and a new episode begins. The agent's action space is defined by four possible actions (up, down, left and right), each of which moves the agent one cell in the corresponding direction with probability 1. If the agent chooses an action that would move it out of the grid world then it remains where it is for that time step. Table 1 shows the hyper-parameter values used in all grid world simulations.

2.3.2. Cart–Pole

In addition to the grid world task we also evaluated CTDL on the Cart–Pole environment as provided by the OpenAI Gym (Brockman et al., 2016). The Cart–Pole problem consists of a cart with a pole attached by a single un-actuated joint. The goal of the agent is to control the velocity of the cart on a linear friction-less track so that the pole stays up-right. The state observed by the agent is made up of four values which correspond to the position of the cart $[-4.8, 4.8]$, the velocity of the cart $[-\infty, \infty]$, the angle of the pole $[\sim -41.8, \sim 41.8]$ and the velocity of the end of the pole $[-\infty, \infty]$. Two actions are available to the agent; push the

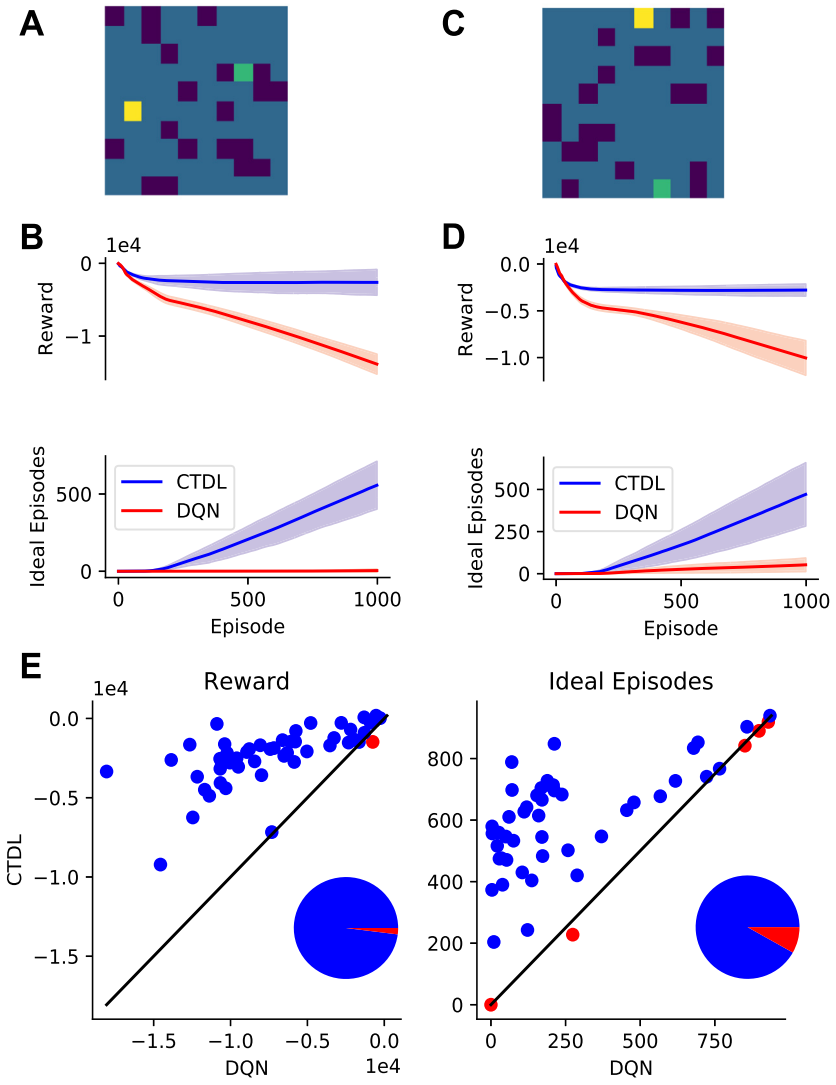

Fig. 1. A: First example grid world, dark blue cells represent negative rewards (−1), the green cell represents the goal (+1) and the yellow cell represents the agents starting position. **B:** Performance of CTDL and DQN on the first example gird world in terms of cumulative reward and 'ideal' episodes over the course of learning. An 'ideal' episode is an episode where the agent reached the goal location without encountering a negative reward. Both CTDL and DQN were run 30 times on each maze. **C:** Second example gird world. **D:** Performance of CTDL and DQN on the second example grid world. **E:** Scatter plots comparing the performance of CTDL and DQN on 50 different randomly generated grid worlds. Both CTDL and DQN were run 30 times on each maze and the mean value at the end of learning was calculated. Blue points indicate grid worlds where CTDL out-performed DQN and red points indicate grid worlds where DQN out-performed CTDL. The pie charts to the lower right indicate the proportions of blue and red points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

cart left and push the cart right. The agent receives a reward of +1 at every time step and an episode ends either when the angle of the pole is greater than 15 degrees, the cart moves off the screen or the episode length is greater than 500.

2.3.3. Continuous mountain car

To explore the applicability of CTDL to continuous control problems we also performed simulations using the continuous mountain car environment as provided by the OpenAI Gym (Brockman et al., 2016). The continuous mountain car environment is a 2D problem consisting of a car that starts in-between two hills. The goal of the agent is to drive the car to the top of the right-hand hill. This problem is complicated by the fact that the cars engine has insufficient power to drive straight up the hill. The agent therefore needs to learn to drive forwards and backwards in order to gain momentum and traverse the hill. The state observed by the agent is defined by the cars position $[-1.2, 0.6]$ and velocity $[-0.07, 0.07]$. Importantly the action space is continuous; the agent must choose to apply a force between 1 and −1 to the car at each time step. The agent receives a reward of +100 for reaching the target location but also receives a negative reward that is equal to the squared sum of the actions it has chosen. An episode terminates either when the car reaches the target location or the episode length is greater than 1000.

3. Results

In our first simulation we compare CTDL to the standard DQN described by Mnih et al. (2015) on a range of grid worlds. Both CTDL and DQN utilize the same DNN architecture (see Table 2 for hyper-parameter values) but there are two key differences between the two approaches. Firstly, a standard DQN stores a memory buffer of size N that is used to replay past experiences whereas CTDL relies on the contents of a SOM for replay. For our simulations we set the memory buffer size M of the DQN

Table 2
Hyper-parameter values used for the DNN component of DQN and CTDL in the grid world simulations.

Parameter	Value	Description
L	3	Number of layers
U	[128, 128, 4]	Number of units
C	10,000	Number of steps before updating the target network
B	32	Batch size for training
λ	.00025	Learning rate for RMSProp
κ	.95	Momentum for RMSProp
ϕ	.01	Constant for denominator in RMSProp

Table 3
Hyper-parameter values unique to CTDL. τ_η, τ_δ, σ, σ_c, α and ρ were selected by using a random grid search on a single grid world.

Parameter	Value	Description
U	36	Number of units in SOM
τ_η	10	Temperature for calculating η
τ_δ	1	Temperature for calculating δ
σ	.1	Standard deviation of the SOM neighbourhood function
σ_c	.1	Constant for denominator in SOM neighbourhood function
α	.01	Learning rate for updating the weights of the SOM
ρ	.9	Learning rate for updating the Q values of the SOM

to 100,000 while the size of the SOM was set to 36 units. This represents a significant decrease in memory resources between the two approaches. The second key difference is that a standard DQN only uses the DNN for calculation of Q values whereas CTDL also incorporates the predictions of a SOM. This allows CTDL to utilize the benefits of a 'hippocampal' learning system during decision making, namely pattern-separated memories and larger learning rates. Hyper-parameter values specific to CTDL can be seen in Table 3. Both models learned from 1000 episodes, with a maximum episode length of 1000. The probability of randomly selecting an action ϵ was linearly decreased from 1.0 to 0.1 over the first 200 episodes. The discount factor for future rewards was set to 0.99 for all simulations.

Fig. 1 demonstrates the results of the two approaches on a random selection of grid worlds. CTDL outperforms the DQN in terms of cumulative reward and the cumulative number of 'ideal' episodes. An ideal episode is classified as an episode where the agent avoids all negative rewards and reaches the positive reward. These findings suggest that the inclusion of a second 'hippocampal' system, which explicitly contributes to the calculation of Q values, is beneficial in our simple grid world task. This gain in performance is achieved at a much lower cost in terms of memory resources. Fig. 2 shows an example maze along with the weights of each unit in the SOM and the location they represent in the maze at the end of learning.

To improve our understanding of the mechanisms underlying CTDLs performance we isolated the contribution of the SOM to the calculation of the Q values from the replaying of the contents of the SOM to the DNN. Fig. 3A shows the performance of CTDL both with and without replay. CTDLs performance was only marginally reduced by the removal of replay suggesting that the improvements over the DQN are due to the contribution of the SOM to the calculation of Q values. A key component of CTDL is the updating of the SOM using the TD error from the DNN. To investigate the importance of this interaction, we compared CTDL to a version of CTDL that did not update the SOM based on the TD error from the DNN. This was achieved by setting the learning rate of the SOM to 0 so that the weights β_u were not updated during learning. Fig. 3B shows the results of this comparison. Removal of the interaction between the DNN and the SOM via the TD signal had a significant impact on the performance of CTDL, suggesting that it is a critical component of the model.

One interpretation of these results is that the SOM is able to store and use experiences that violate generalizations made by the DNN and that this confers a significant advantage during learning. To test this hypothesis we ran CTDL and DQN on three new mazes (Fig. 4). The first maze had no negative rewards between the start and goal locations and the agent simply had to travel directly upwards. We predict that such a maze should favour the DQN because it can rely upon the generalization that an increase in y corresponds to an increase in expected return. The second and third mazes introduced negative rewards that violate this generalization. For these mazes we predict that CTDL should perform better because it can store states that violate the generalization in its SOM and when these states are re-visited CTDL can consult the Q values predicted by the SOM. Fig. 4 shows the results of CTDL and DQN on these three mazes. To help visualize the locations encoded by the SOM we reduced the SOM size to 16 units. The results provide support for our predictions, with the DQN out-performing CTDL in the first maze but not in the second and third mazes. Interestingly, over the course of learning the locations encoded by the SOM appeared to reflect regions of the maze that correspond to violations in the 'move upwards' generalization. We take these findings as evidence that the SOM is encoding states that violate generalizations made by the DNN and that this is responsible for CTDLs improved performance.

If the SOM does encode states that violate generalizations made by the DNN, then this should translate to improved behavioural flexibility in the face of environmental changes. For example if an obstacle appears in one of the grid worlds then this should lead to a large TD error and instruct the SOM to encode the position of the obstacle using its large learning rate. Subsequently, since the SOM keeps track of action values independently from the DNN, CTDL should be able to quickly adapt its behaviour in order to avoid the obstacle. To investigate this hypothesis we ran CTDL and DQN on the grid world in Fig. 4A immediately followed by the grid world in Fig. 4C. Fig. 5 shows the results of these simulations. As previously described, the DQN out-performed CTDL on the first grid world in terms of cumulative reward and the number of ideal episodes. Switching to the second grid world impacted the performance of both the DQN and CTDL. However, this impact was more pronounced for the DQN, with a larger decrease in cumulative reward and a plateauing of the number of ideal episodes. This suggests that CTDL is better equipped to handle changes in the environment. As before the locations encoded by the SOM appeared to reflect states immediately preceding the obstacle. This is consistent with the notion that the TD error from the DNN allows the SOM to identify regions that violate the generalizations made by the network and subsequently improve learning.

One of the strengths of RL algorithms is that they can be applied to a wide array of tasks. If one can describe a task using a state space, an action space and a reward function then often it can be solved using RL techniques, especially if the states satisfy the Markov property. We therefore wanted to investigate whether the performance of CTDL was specific to grid worlds or whether it could be applied to other tasks. We chose to test CTDL on the Cart–Pole and Continuous Mountain Car environments from OpenAI Gym.

We chose to test CTDL on the Cart–Pole environment from OpenAI Gym. We chose the Cart–Pole environment because it is a common benchmark task in the RL literature and it involves a continuous state space, unlike the discrete state space of the grid world environments. The parameter values used for all Cart–Pole simulations were the same as in the grid world simulations with two exceptions. Firstly, the number of time steps C between

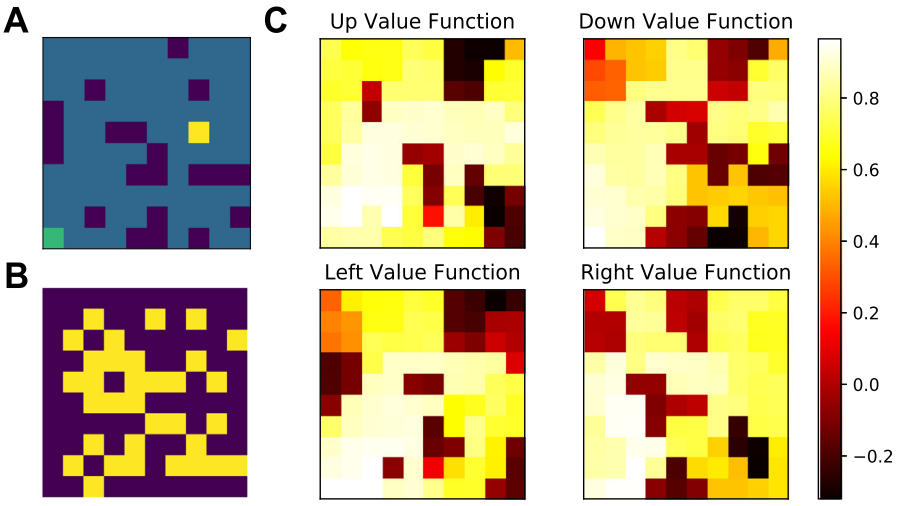

Fig. 2. A: Randomly generated grid world, dark blue cells represent negative rewards (−1), the green cell represents the goal (+1) and the yellow cell represents the agents starting position. **B:** Image showing the locations encoded by the SOM component of CTDL (yellow cells) at the end of learning in A. **C:** CTDLs value function at the end of learning in A, the value is calculated as the weighted average of the predictions from the SOM and DNN. Each state has four possible values, corresponding to each of the four possible actions (up, down, left and right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

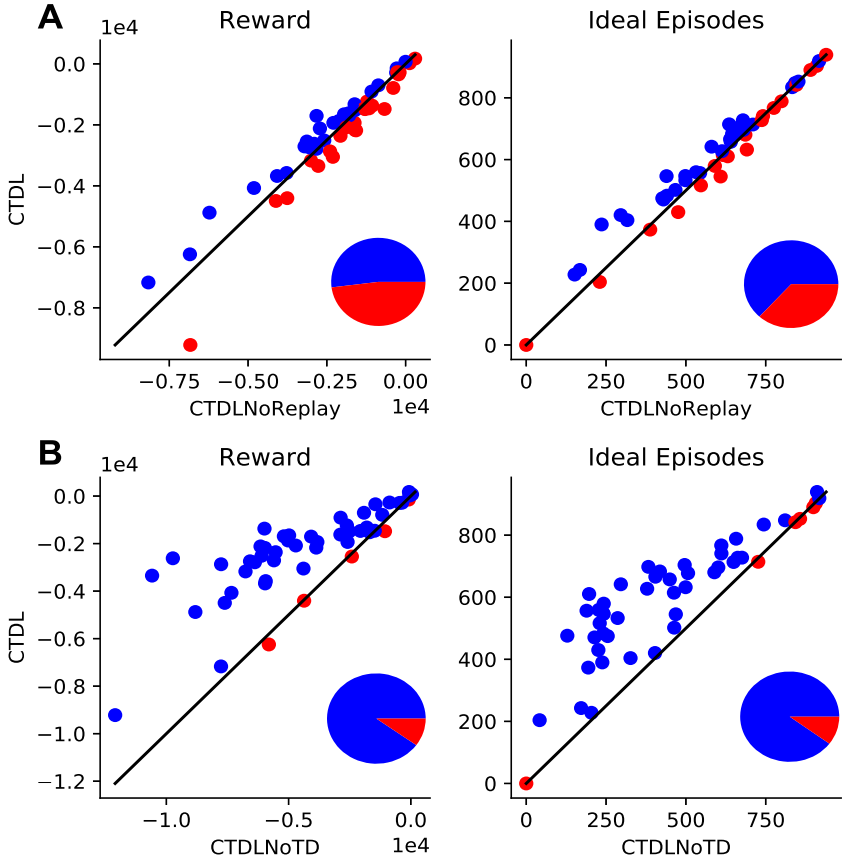

Fig. 3. A: Scatter plots comparing the performance of CTDL and CTDL without replay on 50 different randomly generated grid worlds. Both CTDL and CTDL without replay were run 30 times on each maze. Blue points indicate grid worlds where CTDL out-performed CTDL without replay and red points indicate grid worlds where CTDL without replay out-performed CTDL. The pie chart to the lower right indicate the proportions of blue and red points. **B:** Scatter plots comparing the performance of CTDL and CTDL without TD learning in 50 different procedurally generated grid worlds. Both CTDL and CTDL without TD learning were run 30 times on each maze. Blue points indicate grid worlds where CTDL out-performed CTDL without TD learning and red points indicate grid worlds where CTDL without TD learning out-performed CTDL. The pie charts to the lower right indicate the proportions of blue and red points. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

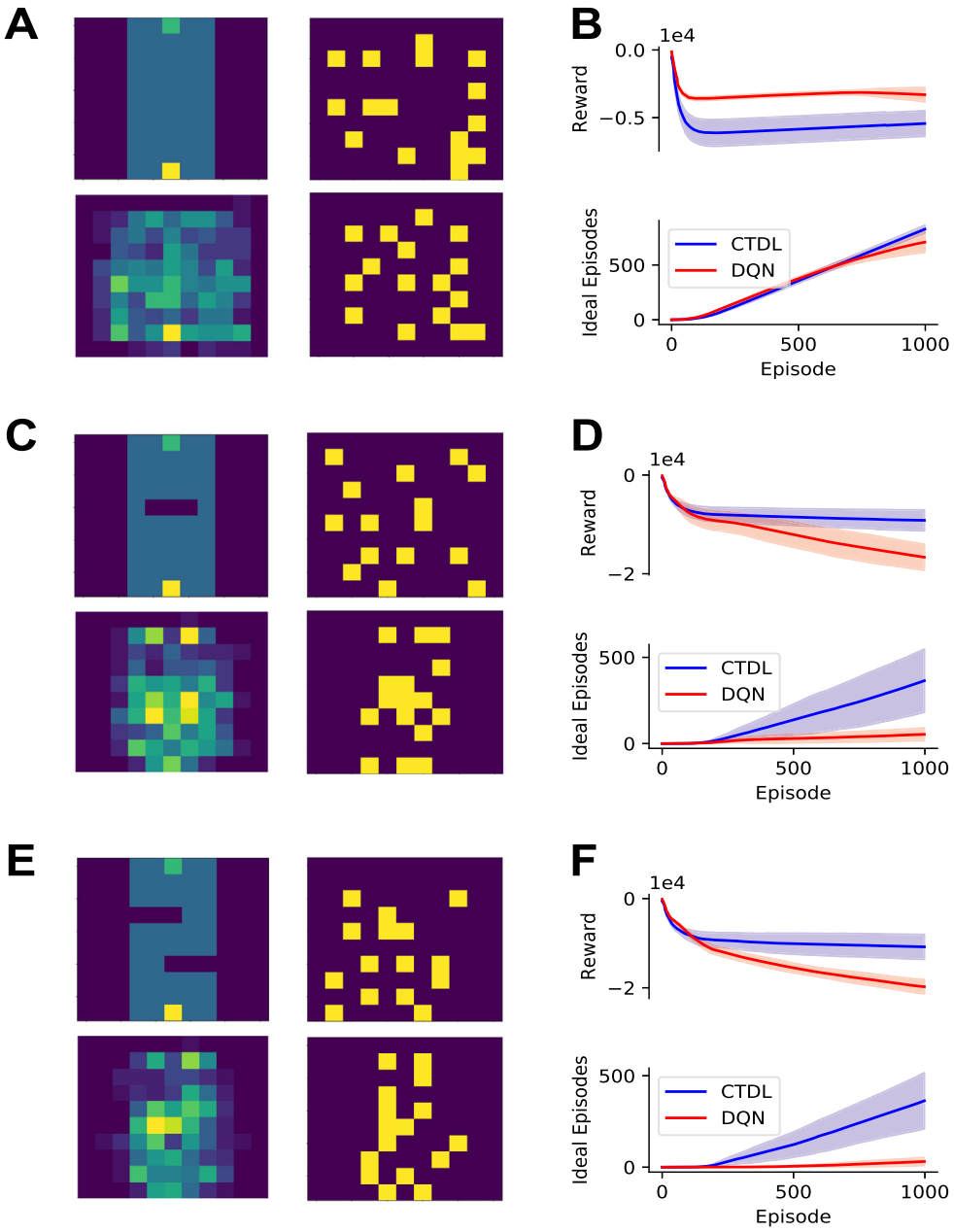

Fig. 4. A: *Top-Left:* Grid world where the agent only has to travel upwards to reach the goal. Dark blue cells represent negative rewards (−1), the green cell represents the goal (+1) and the yellow cell represents the agents starting position. *Bottom-Left:* Locations encoded by the SOM component of CTDL at the end of learning, results are averaged over 30 runs. *Top-Right:* Locations encoded by the SOM component of CTDL at the start of learning for a single run. *Bottom-Right:* Locations encoded by the SOM component of CTDL at the end of learning for a single run. **B:** The performance of CTDL and DQN on the grid world from A in terms of cumulative reward and 'ideal' episodes. The solid line represents the mean and the shaded region represents the standard deviation. **C:** Same as A but an obstacle is introduced, in the form of negative rewards, that the agent must circumnavigate. **D:** The performance of CTDL and DQN on the grid world from C. **E:** Same as C but with two obstacles for the agent to circumnavigate. **F:** The performance of CTDL and DQN on the grid world from E. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

updates of the target network was changed to 500 in order to account for the shorter episodes experienced in the Cart–Pole task. Secondly, the size of the SOM was increased from 36 units to 225 units, which is still considerably smaller than the size of the replay buffer used by the DQN (100,000).

An important component of CTDL is the calculation of the euclidean distance between the current state s_t and the weights of each unit β_u. In the case of the Cart–Pole task this will cause the velocity values in the state representation to dominate the distance calculations because their values cover a much greater range. To account for this we maintain an online record of the largest and smallest values for each entry in the state representation and use these values to normalize each entry so that they lie in the range [0, 1]. This ensures that each entry in the state representation contributes equally to any euclidean distance calculations.

Fig. 6 shows the results of both CTDL and DQN on the Cart–Pole task. While the DQN appeared to learn faster than CTDL, it did so with greater variance and the stability of the final solution was poor. In comparison, CTDL learnt gradually with less variance and there were no significant decreases in performance. These results demonstrate that CTDL can be applied to continuous state

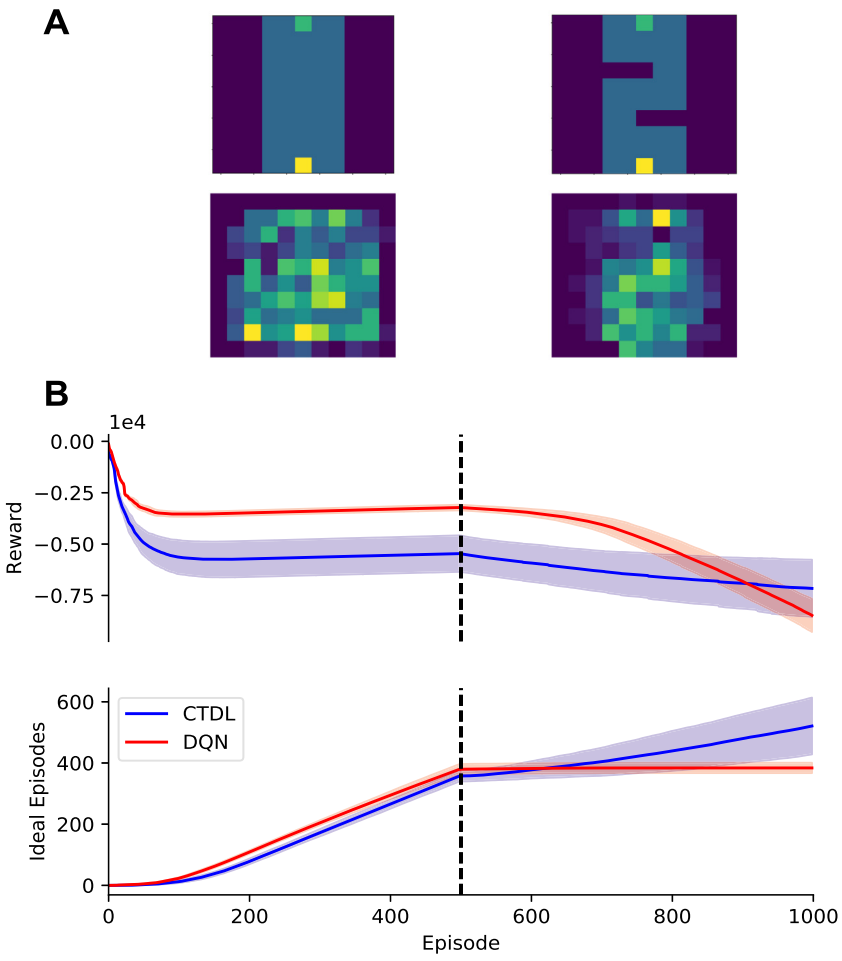

Fig. 5. A: *Top-Left:* First grid world presented to the agent for 500 episodes. *Bottom-Left:* Locations encoded by the SOM component of CTDL at the end of learning in the first grid world, results are averaged over 30 runs. *Top-Right:* Second grid world presented to the agent for 500 episodes immediately after the first grid world. *Bottom-Right:* Locations encoded by the SOM component of CTDL at the end of learning in the second grid world, results are averaged over 30 runs. **B:** The performance of CTDL and DQN on the successive grid worlds from A. The solid line represents the mean and the shaded region represents the standard deviation. The dashed line indicates the change in grid worlds and the introduction of the obstacles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

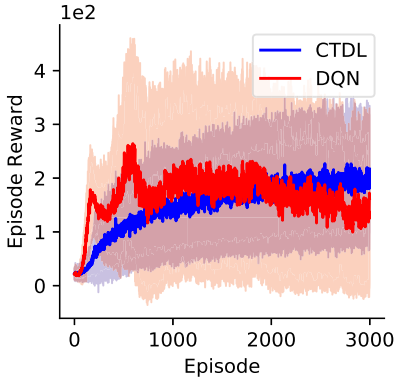

Fig. 6. Episode reward achieved by CTDL and DQN on the Cart–Pole environment over the course of learning. Both CTDL and DQN were run 100 times on the Cart–Pole environment. The solid line represents the mean and the shaded region represents the standard deviation.

problems and is not restricted to discrete grid world problems. They also suggest that CTDL's use of dual learning systems may confer a stability advantage that improves the robustness of learning.

While the Cart–Pole environment uses a continuous state space it restricts the agent to a small discrete action space. We therefore explored whether CTDL could be applied to problems that require a continuous action space as well as state space. To this end we applied CTDL to the Continuous Mountain Car environment from OpenAI Gym. The DNN used in both DQN and CTDL has a single output unit for each action that outputs the Q value for that particular action. This is infeasible for continuous control problems and so a different underlying network architecture is required. A common approach to combining value estimates with continuous control problems is to use an actor-critic framework (Konda & Tsitsiklis, 2000; Sutton, McAllester, Singh, & Mansour, 2000). Under the actor-critic framework, the 'critic' is responsible for calculating value estimates and the 'actor' is responsible for choosing actions and updating the policy based on the values estimated by the critic. The benefit here is that the critic can calculate state values rather than action values and the actor can output a continuous distribution over possible actions.

In our simulations we represent both the actor and critic components as feed-forward neural networks and adopt an Advantage Actor-Critic (A2C) approach (synchronous version of Asynchronous Actor-Critic (A3C) (Mnih et al., 2016)). The hyper-parameters for our A2C implementation can be seen in Table 4. Our implementation of A2C shares many common properties

Table 4
A2C hyper-parameter values used for the continuous mountain car simulations.

Parameter	Value	Description
L_{critic}	3	Number of layers in critic network
U_{critic}	[128, 128, 1]	Number of units in critic network
α_{critic}	.0001	Critic learning rate for Adam
L_{actor}	3	Number of layers in actor network
U_{actor}	[128, 128, 2]	Number of units in actor network
α_{actor}	.00001	Actor learning rate for Adam

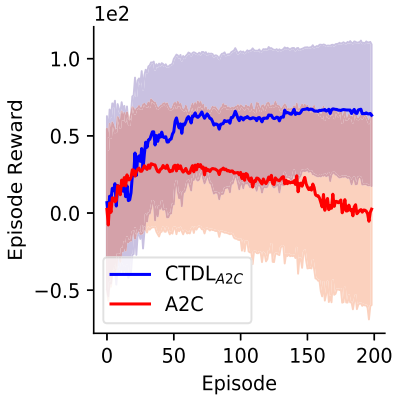

Fig. 7. Episode reward achieved by CTDL$_A$2C and A2C on the Continuous Mountain Car environment over the course of learning. Both CTDL$_A$2C and A2C were run 50 times on the Continuous Mountain Car environment. The solid line represents the mean and the shaded region represents the standard deviation.

with DQN in that it relies upon the slow learning of distributed representations. We therefore hypothesized that the fast pattern-separated learning of CTDL should confer the same advantages to A2C as it did to DQN.

In order to augment A2C with the advantages of CTDL we used the same approach as before except the SOM recorded state value estimates rather than action value estimates. The state value estimates of the SOM were combined with the estimates of the A2C 'critic' using the same weighted sum calculation and the TD error from the 'critic' was used to update the weights of the SOM. A2C is inherently an online algorithm and so weight and value updates were simply applied at each time step in an online fashion with no replay or target networks. We shall denote the CTDL version of A2C as CTDL$_{A2C}$ and a full outline of the algorithm can be seen in Algorithm 2. The hyper-parameters used for CTDL$_{A2C}$ are the same as those in Table 3.

Fig. 7 shows the results of A2C and CTDL$_{A2C}$ on the Continuous Mountain Car task. CTDL$_{A2C}$ outperformed A2C on the Continuous Mountain Car task and also demonstrated much greater stability as training progressed. The high variability in reward obtained is due to the fact that if the agent does not find the target location quickly enough then it will learn to minimize negative rewards by staying still. This suggests that the fast learning of pattern-separated representations in the SOM component of CTDL$_{A2C}$ may allow the agent to either explore more efficiently or better utilize information about states that are rarely visited. In general these results suggest that the dual learning systems of CTDL$_{A2C}$ are advantageous for problems consisting of continuous state and action spaces.

4. Discussion

According to CLS theory, the brain relies on two main learning systems to achieve complex behaviour; a 'neocortical' system that relies on the slow learning of distributed representations and a 'hippocampal' system that relies on the fast learning of pattern-separated representations. Both of these systems project to the striatum, which is believed to be a key structure in the evaluation of states and actions for RL (Houk et al., 1995; Roesch et al., 2009; Schultz, 1998; Schultz et al., 1992; Setlow et al., 2003). Current deep RL approaches have made great advances in modelling complex behaviour, with DNNs sharing several similarities with a 'neocortical' learning system. However these approaches tend to suffer from poor data efficiency and general inflexibility (Lake et al., 2017). The purpose of the present study was to explore how a 'neocortical' and 'hippocampal' learning system could interact within an RL framework and whether CLS theory could alleviate some of the criticisms of deep RL.

Our novel approach, termed CTDL, used a DNN as a 'neocortical' learning system and a SOM as a 'hippocampal' learning system. Importantly the DNN used a small learning rate and distributed representations while the SOM used a larger learning rate and pattern-separated representations. Our approach is novel in that the SOM contributes to action value computation by storing action values independently from the DNN and uses the TD error produced by the DNN to update its state representations. More specifically, the TD error produced by the DNN is used to dynamically set the learning rate and standard deviation of the neighbourhood function of the SOM in an online manner. This allows the SOM to store memories of states that the DNN is poor at predicting the value of and use them for decision-making and learning. Importantly the size of the SOM is smaller than the state space encountered by the agent and so it requires less memory resources than the purely tabular case.

We compared the performance of CTDL to a standard DQN on a random set of 2D grid worlds. CTDL out-performed the DQN on the majority of grid worlds, suggesting that the inclusion of a 'hippocampal' learning system is beneficial and confirming the predictions of CLS theory. Removal of replay between the SOM and DNN appeared to have marginal impact upon the performance of CTDL suggesting that the SOMs contribution to the calculation of action values is the predominant benefit of CTDL. Future work should explore how information from the SOM may be replayed to the DNN in a more principled fashion (e.g. Mattar and Daw (2018)) instead of random sampling. We proposed that the SOM was able to contribute to the calculation of the action values in a targeted manner by using the TD error of the DNN to encode states that the DNN was poor at evaluating. We provided evidence of this by demonstrating that the removal of the TD signal between the DNN and SOM had a negative impact upon the performance of CTDL.

Our interpretation of these results is that, particularly early on in learning, the DNN is able to represent generalizations of the state space while the SOM is able to represent violations of these generalizations. In combination these two systems can then be used to formulate policies in both a general and specialized manner. We tested this hypothesis by presenting CTDL and DQN with a grid world consisting of a general rule and two other grid worlds consisting of violations of this rule. As our interpretation predicted, CTDL out-performed the DQN when violations of the general rule were present, presumably because the SOM was able to store states that were useful for circumnavigating these violations. This hypothesis was further supported by a simulation that ran both CTDL and DQN on sequential grid worlds. CTDL appeared to be better equipped to deal with the change in environment compared to the DQN. In addition, the SOM component of CTDL encoded states close to the change in the environment, providing further evidence of its ability to represent violations of predictions.

This ability of the SOM to encode violations of the generalizations made by the DNN has interesting parallels to imaging work

Algorithm 2 - CTDL$_{A2C}$. Highlighted lines are unique to CTDL$_{A2C}$ when compared to A2C

Initialize SOM weights β according to a standard normal distribution
Initialize SOM state-values $V^{SOM} = 0$
Initialize Critic V^{A2C} with random weights θ^V
Initialize Actor π with random weights θ^π
for $e = 1, E$ **do**
  for $t = 1, T$ **do**
    Observe current state s_t and reward r_t
    Retrieve SOM unit u_t that is closest to s_t
    $u_t = \arg\min_u ||\beta_u - s_t||^2$
    Calculate weighting η based on distance
    $\eta = \exp(-||\beta_{u_t} - s_t||^2/\tau_\eta)$
    Calculate $V(s_t)$ as weighted average of SOM and Critic values
    $V(s_t) = \eta V^{SOM}(u_t) + (1-\eta)V^{A2C}(s_t;\theta^V)$

$$\text{set } y_t = \begin{cases} r_t & \text{if episode is over} \\ r_t + \gamma V(s_t) & \text{otherwise} \end{cases}$$

    Calculate the advantage/TD error $A(s_{t-1}, a_{t-1}) = y_t - V^{A2C}(s_{t-1};\theta^V)$
    Update the Actor parameters θ^V
    $\theta^V \leftarrow \theta^V + \alpha_{actor} A(s_{t-1}, a_{t-1}) \nabla_{\theta^\pi} log(\pi(a_{t-1}|s_{t-1};\theta^\pi))$
    Update the Critic parameters θ^π
    $\theta^\pi \leftarrow \theta^\pi + \alpha_{critic} A(s_{t-1}, a_{t-1}) \nabla_{\theta^V} V^{A2C}(s_{t-1};\theta^V)$
    Calculate δ based on the TD error produced by the Critic
    $\delta = \exp(|A(s_{t-1}, a_{t-1})|/\tau_\delta) - 1$
    Calculate the neighbourhood function based on u_{t-1}
    $T_{u_j,u_{t-1}} = \exp(-||l_{u_j} - l_{u_{t-1}}||^2/2(\sigma_c + (\delta * \sigma)))$
    Update the weights β of SOM
    $\Delta\beta_{ji} = \alpha * \delta * T_{u_j,u_{t-1}}(s_{t-1,i} - \beta_{ji})$
    Update the state value $\Delta V^{SOM}(u_{t-1}) =$
    $\rho * \eta_{t-1} * (y_t - V^{SOM}(u_{t-1}))$
    Sample action from Actor $a_t \sim \pi(a_t|s_t;\theta^\pi)$
    If the goal has been reached then **break** and end episode
  end for
end for

in rodents demonstrating that CA3 neurons appear to encode decision points in T-mazes that are different from the rodents current position (Johnson & Redish, 2007). Such decision points can be viewed as obstacles or important deviations from the animals general direction and we therefore predict that they would be encoded by the SOM component of CTDL. In the future, application of CTDL to other reinforcement learning tasks may provide testable predictions about the regions of the state space that should be encoded by the hippocampus. In addition, the fact that the SOM encode states close to obstacles in order to account for changes in the environment suggests that the hippocampus may be important for adapting to changes in the environment and is consistent with recent studies that have implicated the hippocampus in reversal learning (Dong et al., 2013; Vila-Ballo et al., 2017).

To investigate the generality of CTDL we also applied it to the Cart–Pole and Continuous Mountain Car problems. The Cart–Pole problem is fundamentally different to the grid world problem because the state space observed by the agent is continuous. We found that in comparison to the DQN, the learning of CTDL was more gradual but also more robust. This is perhaps a surprising result given that the DQN has a perfect memory of the last 100,000 state transitions whereas CTDL has no such memory. Indeed, one would expect the SOM component of CTDL to have less of an effect in continuous state spaces because generalization from function approximation becomes more important and the probability of re-visiting the same states decreases. With this being said, Blundell et al. (2016) demonstrated that even when the probability of re-visiting the same state is low, episodic information can still be useful for improving learning. Generalization of episodic information in CTDL is likely controlled by the temperature parameter τ_η that scales the euclidean distance between the states and the weights of the SOM units.

The Continuous Mountain Car problem consists of both a continuous state and action space. In order to apply deep RL to the Continuous Mountain Car problem we used an A2C architecture, with two separate DNNs representing an actor and critic respectively. As with the original implementation of CTDL, we augmented A2C with a 'hippocampal' learning system in the form of a SOM and termed the resulting algorithm CTDL$_{A2C}$. CTDL$_{A2C}$ both outperformed the standard A2C approach and demonstrated more robust learning with no substantial decreases in performance. A defining feature of the Continuous Mountain Car problem is that the agent will learn not to move unless it experiences the positive reward of the target location and then utilizes this information efficiently. It is possible that the addition of a learning system that quickly learns pattern-separated representations helps to alleviate this problem by storing rare and surprising events in memory and incorporating them into value estimates, rather than taking a purely statistical approach. More generally, these results demonstrate the applicability of CTDL to continuous control problems and further highlight the benefits of using TD error to inform the storage of episodic information.

The reduced benefit of CTDL on the Cart–Pole problem compared to the Grid World and Continuous Mountain Car problems may allude to interesting differences in task requirements. In particular, both the Grid World and Continuous Mountain Car problems appear to rely on rare discrete events that are highly informative for learning a policy e.g. both tasks involve a goal location. In comparison, the Cart–Pole task relies on a range of rewarded events or states to inform the policy and so the utilization of episodic information may be less valuable. From a biological perspective, it is perhaps unsurprising that CTDL performs better on Grid World problems given that they represent spatial navigation tasks which are thought to heavily recruit the hippocampus in biological agents (Burgess, Maguire, & Keefe, 2002). In comparison, the Cart–Pole problem can be seen as a feedback-based motor control task which involves learning systems such as the cerebellum in addition to any cortical-hippocampal contributions. CTDL may therefore represent a useful empirical tool for predicting the utilization of hippocampal function in biological agents during RL tasks.

Future work will need to investigate whether the increased robustness and performance of CTDL in continuous state and action spaces is a general property that extends to more complex domains. In particular, it would be of interest to run CTDL on maze problems such as ViZDoom (Kempka, Wydmuch, Runc, Toczek, & Ja, 2016), which are rich in visual information. Indeed, deep RL approaches using convolutional neural networks are at the forefront of RL research and these could be easily incorporated into the CTDL approach. In the case of ViZDoom, each state is represented by a high-dimensional image and so the generalization capabilities of a DNN are crucial. From a biological perspective, it is worth noting that the hippocampus operates on cortical inputs that provide latent representations for episodic memory. This is captured in 'model-free episodic control', which relies on an embedding function to construct the state representation for episodic memory (Blundell et al., 2016; Pritzel et al., 2017). An embedding function therefore represents a biologically plausible method of scaling CTDL up to complex visual problems such as VizDoom. The embedding function could be pre-trained in an unsupervised manner or sampled from the DNN component of CTDL. We leave this interesting avenue of research to future work.

In addition to relatively low complexity, one consistent feature of the tasks presented in the present study was a low degree of stochasticity. As with discrete state spaces, low stochasticity means that events re-occur with high probability and the episodic component of CTDL can exploit this. It is likely that in more stochastic environments the benefits of CTDL will be reduced as the DNN is required to generalize over several outcomes. It is therefore an open question how well CTDL will perform on tasks that have a high degree of stochasticity, which are also supposedly harder for biological agents.

One interesting element of CTDL that was not explored in the present study was the temporal evolution of pattern-separated representations in the SOM. Logically as the DNN improves its ability to evaluate the optimal value function its TD errors should reduce in magnitude and therefore free up the SOM to represent other episodic memories. If part of the environment changes then a new episodic memory will form based on the new TD error and it will remain in episodic memory until the 'neocortical' learning system has learnt to incorporate it. CTDL therefore suggests that the transfer of information from the hippocampus to neocortex is based to the 'need' for an episodic memory as encoded by TD errors. This can be viewed as a form of 'consolidation' whereby memories stored in the hippocampus are consolidated to the neocortex over time (Olafsdottir et al., 2018).

As a concluding remark, we have only demonstrated the benefits of a 'hippocampal' learning system from a purely model-free perspective. A growing body of research however, is implicating the hippocampus in what has historically been considered model-based behaviour. For example, it has been proposed that the hippocampus encodes a predictive representation of future state occupancies given the current state of the agent (Stachenfeld, Botvinick, & Gershman, 2017). This predictive representation has been termed the Successor Representation (SR), and has been shown to imbue agents with model-based behaviour using model-free RL mechanisms in a range of re-evaluation tasks (Russek, Momennejad, Botvinick, Gershman, & Daw, 2017). Therefore the inclusion of a 'hippocampal' learning system may have additional benefits above and beyond those demonstrated by CTDL.

5. Conclusions

Taken together, we believe that our work highlights CTDL as a promising avenue for achieving complex, human-like behaviour and exploring RL within the brain. Having a 'neocortical' and 'hippocampal' learning system operating in parallel conferred a learning advantage over a single 'neocortical' system. This advantage was attributable to two main properties of CTDL. Firstly, both the 'neocortical' and 'hippocampal' system contributed to the calculation of action values for decision-making, with the arbitration between the two dependent on the memory content of the 'hippocampal' system. Secondly, the contents of the 'hippocampal' system were dynamically updated using the TD error from the 'neocortical' system. This allowed the 'hippocampal' system to target regions of the state space that the 'neocortical' system was poor at evaluating or that violated generalizations made by the 'neocortical' system.

These key properties of CTDL represent promising avenues for future research both computationally and empirically. From a computational perspective, it will be interesting to explore how embedding functions can be utilized to reflect the fact that the hippocampus receives latent representations from cortical areas as input. This may be a key component for scaling up CTDL to complex problems with high dimensional state spaces. With respect to future empirical work, CTDL can be used to make predictions about which tasks should utilize the hippocampus and which regions of the state space should be encoded by it. CTDL also predicts that TD errors should promote the formation of episodic memories in the hippocampus and so we highlight this as a key area for further investigation and clarification.

Acknowledgements

This work was funded by the UK Biotechnology and Biological Sciences Research Council (BBSRC). We thank NVIDIA for a hardware grant that provided the Graphics Processing Unit (GPU) used to run the simulations.

Code

All source code can be found at:
https://github.com/SamBlakeman/CTDL

References

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment : An evaluation platform for general agents. (pp. 253–279). arXiv 47, arXiv:arXiv:1207.4708v2.

Blundell, C., Pritzel, A., & Rae, J. (2016). Model-free episodic control. arXiv arXiv:arXiv:1606.04460v1.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-nelson, Z., Blundell, C., & Hassabis, D. (2019). Reinforcement learning , fast and slow. *Trends in Cognitive Sciences*, 1–15.

Bray, S., & Doherty, J. O. (2007). Neural coding of reward-prediction error signals during classical conditioning with attractive faces. *Journal of Neurophysiology*, *97*(4), 3036–3045.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). OpenAI gym. (pp. 1–4). arXiv arXiv:arXiv:1606.01540v1.

Burgess, N., Maguire, E. A., & Keefe, J. O. (2002). The human hippocampus and spatial and episodic memory. *Neuron*, *35*(4), 625–641.

Doherty, J. P. O., Dayan, P., Friston, K., Critchley, H., & Dolan, R. J. (2003). Temporal difference models and reward-related learning in the human brain. *Neuron*, *38*(2), 329–337.

Dong, Z., Bai, Y., Wu, X., Li, H., Gong, B., Howland, J. G., et al. (2013). Neuropharmacology Hippocampal long-term depression mediates spatial reversal learning in the Morris water maze. *Neuropharmacology*, *64*, 65–73.

François-lavet, V., Henderson, P., Islam, R., Bellemare, M. G., François-lavet, V., Pineau, J., et al. (2018). An introduction to deep reinforcement learning. arXiv arXiv:arXiv:1811.12560v2.

Gershman, S. J., & Daw, N. D. (2017). Reinforcement learning and episodic memory in humans and animals : An integrative framework. *Annual Review of Psychology*, *68*, 101–128.

Houk, J. C., Adams, J. L., & Barto, A. G. (1995). A model of how the basal ganglia generate and use neural signals that predict reinforcement. *Computational Neuroscience. Models of Information Processing in the Basal Ganglia*, 249–270.

Johnson, A., & Redish, D. A. (2007). Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, *27*(45), 12176–12189.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Ja, W. (2016). ViZDoom : A doom-based AI research platform for visual reinforcement learning. arXiv arXiv:arXiv:1605.02097v2.

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 10–12, arXiv:1607.07086.

Kumaran, D., Hassabis, D., & McClelland, J. L. (2016). What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends in Cognitive Sciences*, *20*(7), 512–534.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *The Behavioral and Brain Sciences*, *40*.

Lee, D., Seo, H., & Jung, M. W. (2012). Neural basis of reinforcement learning and decision making. *Annual Review of Neuroscience*, *35*, 287–308.

Mattar, M. G., & Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, *21*(11), 1609–1617.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*(3), 419–457.

Mcclure, S. M., Berns, G. S., & Montague, P. (2003). Temporal prediction errors in a passive learning task activate human striatum. *Neuron*, *38*(2), 339–346.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. *48*, arXiv arXiv:arXiv:1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, *53*(3), 139–154.

Olafsdottir, F. H., Bush, D., & Barry, C. (2018). Review the role of hippocampal replay in memory and planning. *Current Biology*, *28*(1), 37–50.

Pennartz, C. M. A., Ito, R., Verschure, P. F. M. J., Battaglia, F. P., & Robbins, T. W. (2011). The hippocampal – striatal axis in learning, prediction and goal-directed behavior. *Trends in Neurosciences*, *34*(10), 548–559.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., et al. (2017). Neural episodic control. arXiv arXiv:arXiv:1703.01988v1.

Roesch, M. R., Singh, T., Brown, P. L., Mullins, S. E., & Schoenbaum, G. (2009). Rats deciding between differently delayed or sized rewards. *Journal of Neuroscience*, *29*(42), 13365–13376.

Russek, E. M., Momennejad, I., Botvinick, M. M., Gershman, S. J., & Daw, N. D. (2017). Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS Computational Biology*, *13*(9), 1–35.

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, *80*(1), 1–27.

Schultz, W. (2016). Dopamine reward prediction error coding. *Dialogues in Clinical Neuroscience*, *18*(1), 23–32.

Schultz, W., Apicella, P., Scarnati, E., & Ljungberg, T. (1992). Neuronal activity in monkey ventral striatum related to the expectation of reward. *Journal of Neuroscience*, *12*(12), 4595–4610.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*(5306), 1593–1599.

Setlow, B., Schoenbaum, G., & Gallagher, M. (2003). Neural encoding in ventral striatum during olfactory discrimination learning. *Neuron*, *38*(4), 625–636.

Stachenfeld, K. L., Botvinick, M. M., & Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, *20*(11), 1643–1653.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (p. 1). Cambridge: MIT press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation richard. *Advances in Neural Information Processing Systems*, *12*, 1057–1063, arXiv:1706.06643.

Vila-Ballo, A., Mas-Herrero, E., Ripolles, P., Simo, M., Miro, J., Cucurell, D., et al. (2017). Unraveling the role of the hippocampus in reversal learning. *Journal of Neuroscience*, *37*(28), 6686–6697.